Probabilistic Music Generator
Requirements Capture

Group 258

Bryce Johnson
David Loken
Chad Swenson

ECE 405
20 January 2006

Introduction

Bob is a student of music at a university and is in the process of taking a difficult test. His job over the next few years is to learn as much as possible about how to play particular instruments, study the history of music, and to simply learn how music works. The latter is a hard question, how does music work? Bob wished he knew at this moment. He is taking a test in a music theory class and is stuck. The idea of this part of the test is to make chord progressions. To make these progressions he learned certain 'rules' that will aid him in this. These 'rules' are supposed to make a type and style of music sound 'good'. Bob has two choices that he hopes will work. He flips a coin and writes out the first chord he was contemplating. His sweat breaks for a moment as continues on with the test.

Bob guessed right. Actually, it turns out, both chords were right. There is a difference in how they sound, but they both seem to fit. Bob used a coin to make his pick[1]. Do we even need Bob at all in this situation? Turn Bob into a computer, create algorithms out of these musical 'rules', and throw in probability and we have system that can create chord progressions. All you need to do is input the first note.

In our project we will use this same idea. Through the analysis of other pieces of music and the algorithmic representation of musical rules, we will create a DSP based system that can play music based on probability.

Our project will implement algorithms on a DSP and output randomly generated musical notes that sound pleasing to the ear. This particular problem of generating random notes that sound pleasing has a reasonably sized following throughout the world. There is a good amount of information on the algorithms used to generate this music and how to put notes together that sound good with each other. Development of simple first order Markov chains can be found on the internet with a simple Google search. There have been databases made of classical music arranged into Markov chains, and used to analyze classical music. Simple chains can also be made from any song or songs with a little reading on how to construct them. Second order and higher Markov chains are also being used and developed by enthusiasts and their work can be found on the internet.

There has been a little development as far as making these algorithms generate music as well. One such project that we found included a low powered microprocessor that used first order chains to calculate what notes to use, and then to activate solenoids that played the notes on a Xylophone. This simple solution is a bulky way of doing what we want to do. We want to do something similar, in that we want to generate the notes and music but in the end we hope to have more complex generation of notes and synthesized note outputs.

---

[1] Bob wasn't the first to think of this. This is not unlike "Musikalisches Würfelspiel" (Mozart's Dice Game). This method for making music is based on the chance in rolling dice. (http://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel)

The simple and straight forward requirements of our project are that we build a product that generates random notes in a pleasing way, that is stand alone (does not require a computer to run the code or do computations), and that can output these notes in CD quality output (44.1KHz sampling). These are the requirement developed by us and our advisor.

Previous Work

There are several examples of previous work that our project is going to draw off of.

Hardware
Our project will be using a DSP board based on TI's TMS320F2812 processor that was designed by Dr. Green's DSP scholar team. This board was designed through two semesters of work with around twelve students participating.

Software
The algorithms that we are developing will use Markov chains. Markov chains have been used to analyze, classify, and produce music. There are several other people that have used this approach.

"Intelligent MIDI Sequencing with Hamster Control"
(http://instruct1.cit.cornell.edu/courses/eceprojectsland/STUDENTPROJ/2002to2003/lil2/).

> Levy Lorenzo used hamsters as to control the relative probabilities of the Markov chains that controlled both pitch and rhythm to create music. He used a MIDI Sequencer for musical note output.

"Grammar-Based Music Composition"
(http://journal-ci.csse.monash.edu.au/ci/vol03/mccorm/mccorm.html).

> This gives several good ideas dealing with Markov chains and with several other ideas for probabilistic music. There are examples on how to use Markov Chains and L-systems which we were considering on using in our project. It also discusses some basic things about music that are useful in understanding, such that it is not the pitch of notes that create music but the differences in pitch and the order in which they happen. Most musical pieces involve repetition in them.

"Xylophone 1.0"
(http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2004/mk257/index.html).

> This project used Markov chains to generate random music to play a xylophone. They used a processor to do pitch detection and used algorithms to choose the next note. From there they used a solenoid to strike the appropriate xylophone key.

"L-systems" (http://en.wikipedia.org/wiki/L-system).

This provides a description of what L-systems are and are generally used for. This is an alternative approach to Markov Chains.

Requirements

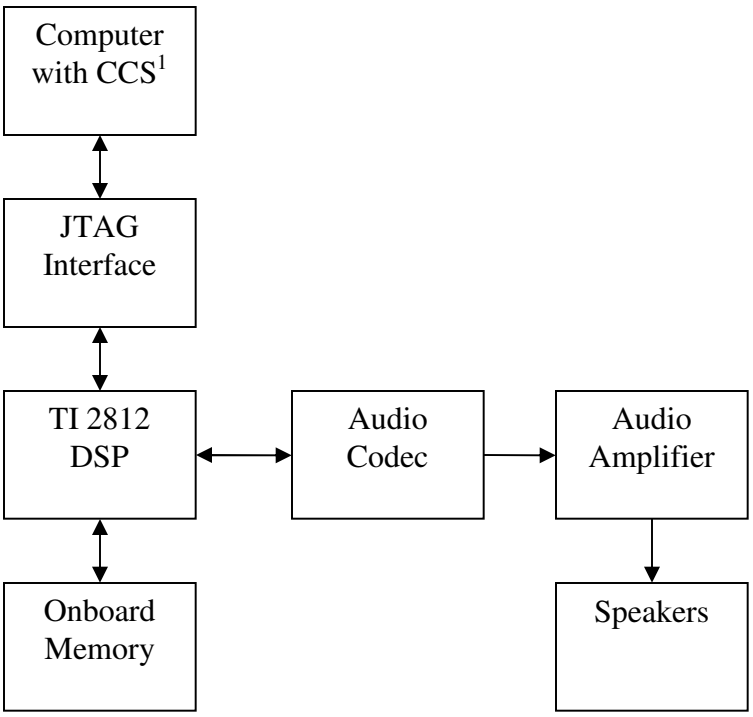
The following are the requirements of our project:

1. The project will be based on the scholar team's TI 2812 DSP processor board.
2. The device must have a JTAG interface (in circuit programming).
3. The device must be able to take in different sets of probability to model different styles of music (e.g. rock, classical, children's' music.)
4. The device must be able to generate notes.
5. The device must be able to generate music via algorithms.
6. The device must be able to output the audio signal it is generating.
7. The output needs to have a 44.1 kHz sampling rate.
8. It must be a standalone device.

Adviser signature:_____ Date:_____

Block Diagram

There are several ways in which we could go about implementing our project. We could build a device that just plays an instrument such as a piano or xylophone. This would do what we want, but we would lose portability. We could also use a computer and just build a program that calculates the notes and then outputs them, but this does not fulfill the stand alone requirement of the client (us) either. The following block diagram is the best way, we believe, to implement the system as the client wants. It shows a stand alone DSP that does the calculations, and sends information to a CODEC to output a signal. The interface for uploading code to the DSP that we will be using is the JTAG emulator.



[1] Code Composer Studio

DSP:

The digital signal processor is a special-purpose embedded processor. DSPs are used to process digital signals extremely fast. For this project the DSP will be used to store and execute a set of algorithms to produce a digital signal that can be can be converted to an audio signal by an audio codec.

JTAG:

JTAG stands for Joint Test Action Group, the group that developed the interface. This interface will allow communication between the digital signal processor and a PC.

On Board Memory

The DSP has a limited amount of memory, so external memory will be required. This memory will hold any data that does not fit on the DSP.

Audio Codec

The audio codec on the DSP board will convert the digital signal form the DSP into a 2 channel analog audio signal.

Options Considered

Most of our board has already been designed and is in the trouble shooting stage of development. This arises from the fact that we are using a board that was developed by the DSP scholar team. Most of our hardware was designed by the DSP scholar group and have to use what they designed. We are in the final process of trouble shooting what they have designed and working on getting their design in working condition.

On-board Memory

There is support for external memory by the DSP so we can use external memory if needed. The DSP supports up to a 2 MB RAM chip so we will be considering this when we run out of on chip memory.

DSP

As far as the DSP goes we are pretty much stuck with what we have. The board is already laid out and the footprint is unique to this processor so we cannot really change the DSP. The DSP we have is the TI 2812. This is a fixed point processor that runs at 150 MHz and has 128Kbytes of flash memory.

Interface

JTAG (Joint Test Action Group) interface-
This interface is available to use and is very useful as far as communication with the DSP goes. The JTAG emulator is already available to us, so this is probably our best choice for an interface. This interface allows two-way communication and allows us to see what the DSP is doing as it steps through the program. It is a very useful tool for troubleshooting our program.

Audio Codec

PCM3003-
We have the ability to change out the 3003 to the 3002. The only difference is that the 3002 is software controlled so we would have to change a little bit of our hardware to set the codec up to communicate with the DSP.

The codec we will be using is the PCM3003 because it is easier to set up and is already on the board. It is setup to communicate with the DSP and has all the jumper pins available to adjust the settings. The PCM3002 would require that we give inputs from the DSP to the codec to set up the codec to sample right and perform right.

Power Supply

- AC Power supply-
  Wall mounted 120VAC to 5VDC converter: This will give us all the power that we will need to supply our device without having to worry about battery life. The disadvantage is that we will be limited to where we can go with the device.
- Batteries-
  Five volt rechargeable battery: With this we could have a portable system. However this limits how long the board can run.

Algorithms

- 1st order Markov Chains-
  This is system based on conditional probability. A new state is based off of the previous state. Using a table we are able to model the probability from one state to all the other states. This would be easily implemented, but it might not produce quality music.
- Higher Order Markov Chains-
  This is similar to 1st order Markov Chains. Instead of a new state being based off just one state, it is based off multiple states. This would produce more phrased music which should sound better. The problem with higher order chains is the complexity in which the chains get and may start taxing our processor while running and having to output data.
- Lindenmayer Systems-
  This is a set of rules and symbols originally used to model the growth processes of plant development. The variables, the start variable, and the rules of the system are what is need to be specified. These variables could easily be converted in to the parameters of music. Using L-systems would be more complex to implement in the DSP then Markov Chains. It would produce similar phrased music as the higher order Markov Chains. This system also has the potential to be taxing on the processor and may cause the calculations to fall behind.

  This choice is a little harder. Ideally we will be able to implement all three as there benefits are all good. In the end we will really be striving to

go with a second order Markov Chain. This will give us better complexity while hopefully not being too taxing on the processor or memory.

Summary

In our project we chose to go with the 2812 DSP due to our collaboration with the scholar team. We will attempt to use as external memory as we will probably run out of on chip RAM. The PCM3003 will be our audio codec because it is controlled by hardware, already on the DSP board and easiest to use. The JTAG interface to the DSP is chosen due to ease of use and availability. We will use an AC power supply for development, batteries will be used if practical after all the development is done to give the desired portability of the device. The Algorithm chosen will be 1st order Markov Chains with the option of extension to 2nd order Markov Chains.

Testing

Each listed requirement will be tested to ensure the completed project works.

1. Put a test program on the scholar team's DSP board to ensure it works. We will run simple programs to test basic functionality. This will include outputting a sine wave and using the on board LEDs to test functionality.
2. The JTAG interface allows communication in both directions so, after attempting to program the DSP, it will be clear if the device is interfacing with the computer or not. This is the only way in which we have to program the DSP Scholar board.
3. We will test whether different sets of probability will produce grossly different sounding music. We will do this in MatLab before we try to do any programming of algorithms on the board.
4. The generation of musical notes can be tested by connecting a speaker to the output and listening.
5. To check the algorithms, we will be testing them in such software as MatLab before they are programmed into the DSP. When they are loaded to the DSP we will be able to do our final tests.
6. To see if the device is outputting a signal we can check the output with an oscilloscope and send the output to a speaker.
7. To test the sampling rate to ensure that is at least 44.1 kHz we will do a few things. First, we will generate signals increasing in frequency until aliasing begins to occur. This should be around ½ the sampling rate which is 22.05 kHz. We will also test the clocks going to the codec to ensure that they are 44.1 kHz.
8. To test that the device is stand alone we will look at the device and see if it is connected to the computer. If it is… it shouldn't be.

Timeline

First Semester

Task Name	Sept.				Oct.				Nov.					Dec.		
weeks	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3
Requirements Capture				X												
Options Considered								X								
Prototype Validation													X			
Presentation															X	
DSP Research																
Algorithm Research																
Audio Amp research																

The main part of the first semester will be researching the DSP and understanding the DSP board that the scholar team has built. This board is not 100% done so it is also our responsibility to help get this board troubleshot and working as soon as possible so that we can start development on the board. We will also be working with MatLab testing algorithms during the first semester. Also we will be working with a less specific board that will allow us to run code on and learn how to use the JTAG interface so that development will be easier in the second semester.

Second Semester

Task Name	Jan			Feb				March					April				May	
weeks	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2
Update time line	G	R	O	U	P													
Research Enclosure				C	H	A	D	/	B	R	Y	C	E					
Research Note Generation			G	R	O	U	P											
Finalize Algorithms	B	R	Y	C	E													
Upload Code and test	C	H	A	D														
Populate and test board					D	A	V	E										
Documentation				G	R	O	U	P							X			
Revise Board Layout			D	A	V	E												
Implementing Algorithms			C	H	A	D	/	B	R	Y	C	E						
Finish up lose ends											G	R	O	U	P	X		
Prepare for Presentations										G	R	O	U	P		X		

Our second semester of work will consist of bringing everything that we had worked on in the first semester together. The algorithms and code be finalized and programmed onto the DSP to be tested. In particular we will focus on revising the DSP board that the scholar team and the senior design team have worked on and getting a second board populated and working. We will be concentrating more heavily on coding than in the first semester. We will be looking at getting all the little pieces of the board to communicate with each other. This includes getting the DSP to send data to the codec and to use the external memory. The last thing, and what really brings everything together, is implementing the algorithms on the DSP and actually performing the main goal of our project which is outputting random music.

Budget

Item	True Value	Acquired Cost
Revision PCB (Four Layer)	$200	$0
2812 DSP (1)	$25	$0
Codec (1)	$8	$0
Voltage Regulator (1)	$4.50	$0
Op-Amps (2)	$4.50	$0
LEDs (20)	$5	$3
Darlington Array (1)	$0.75	$0
Inverter Chip (1)	$0.44	$0
EMI Suppressor (2)	$2.80	$0
Other Inexpensive ICs	$5	$0
Passives (Resistors, Capacitors)	$30	$0
5V Wall-Wart	$10	$10
Buttons (10)	$10	$10
Batteries and charger	$30	$30
LCD (2)	$80	$80
Enclosure	$100	$100
Incidentals	$100	$100
JTAG Interface (510)	$1200	$0
Code Composer	$500	$0
EZ-DSP Board	$325	$0
Totals	$2640.99	$333

We gave an estimate of around $300 during the first semester of development when asked for a budget. This is still roughly the same amount we will need this semester after we broke down the costs that we are expecting and with a little bit of money just in case we run into problems that require a “Franklin fix”. So far the only things that we have need are already here. We have been using Code Composer Studio and the emulator and the current scholar team board.

Advisor Signature: _____ Date: _____